IPA Software Reliability Enhancement Center
Information-technology Promotion Agency, Japan

ESTA
Embedded System Testing Reference Guide

ET-West2013 IPAセミナー

組込みソフトウェアテスト事例集の紹介
組込みソフトウェア開発における品質向上の勧め
[テスト編～事例集～]

2013年6月14日

独立行政法人 情報処理推進機構
技術本部　ソフトウェア高信頼化センター

システムグループ　研究員　　石田　茂
日本電気通信システム株式会社　羽田　裕

Copyright © 2013 IPA, All Rights Reserved

組込みソフトウェアテスト事例集

Practical Testing Examples

SEC Software Reliability Enhancement Center

新刊

Recommendations for Improvement of Quality of Embedded Software Development [w/ Practical Testing Examples]

2012年11月12日発行

書籍：

A5判、100ページ

定価500円（税込）

pdf版：

http://sec.ipa.go.jp/publish/index.html#emb

無償

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
    1. 組込みシステムの不具合発生状況(産業実態把握調査)
    2. 組込みソフトウェアのテスト実態調査結果
2. テスト事例集のコンセプト
3. テスト事例集の特徴
4. テスト事例集の構成
5. 利用方法
6. 事例紹介

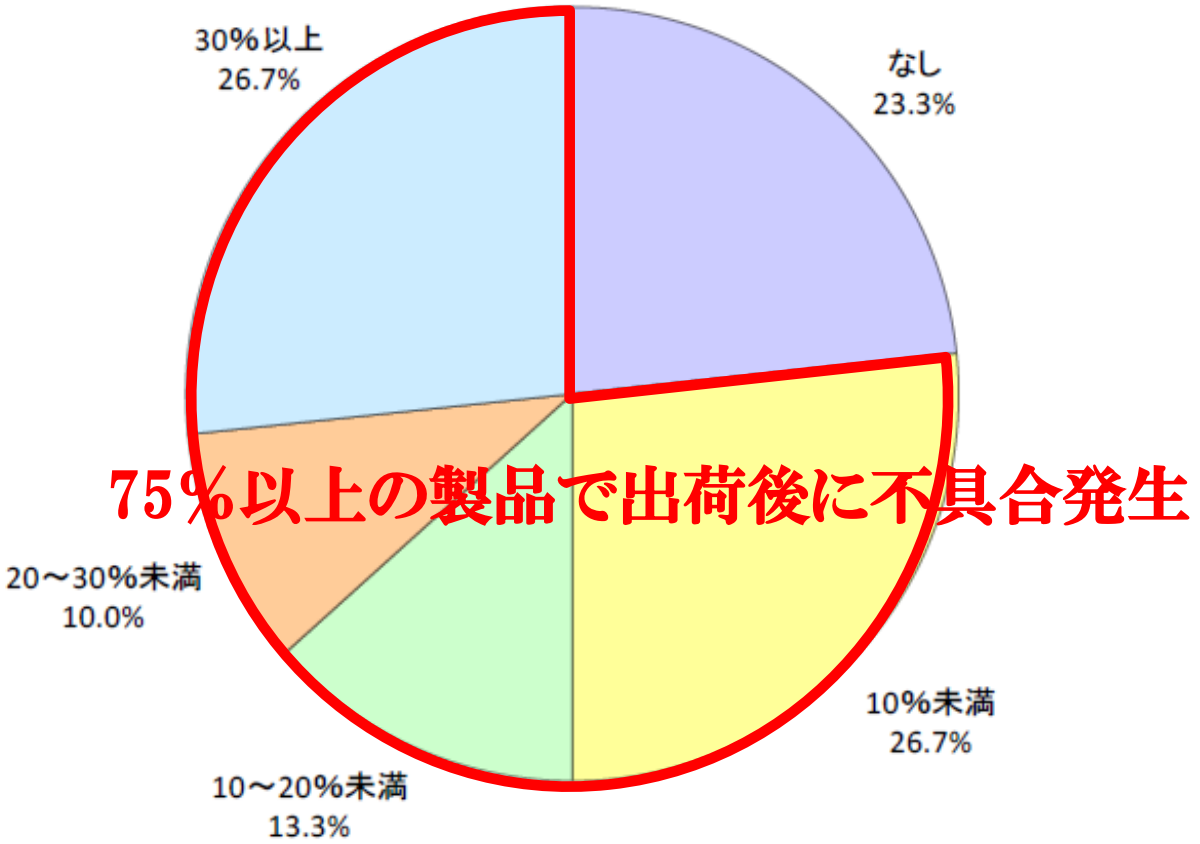
Copyright © 2013 IPA, All Rights Reserved

1．1　組込みシステムの不具合発生状況-1

SEC Software Reliability Enhancement Center

2012年版組込みソフトウェア産業実態把握調査報告書

不具合発生製品率

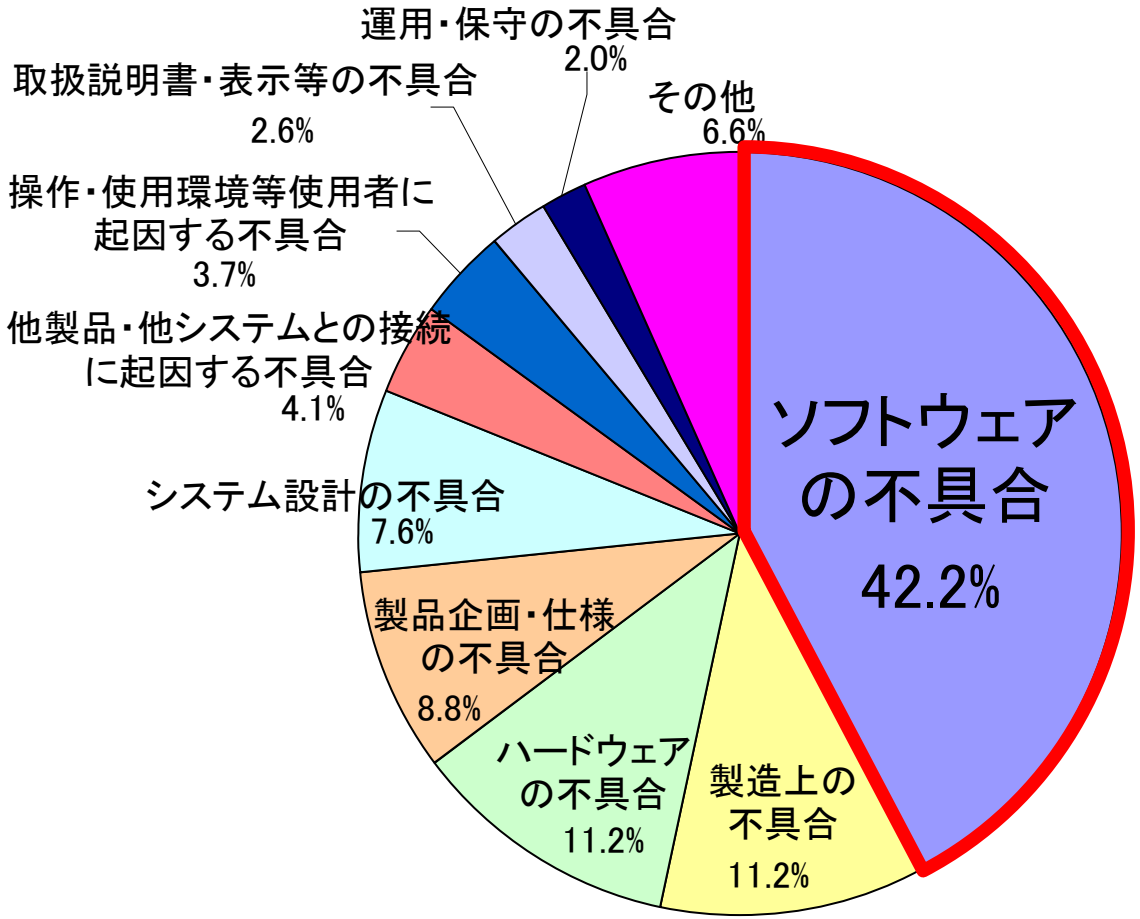


Copyright © 2013 IPA, All Rights Reserved

1．1　組込みシステムの不具合発生状況-2

2011年版組込みソフトウェア産業実態把握調査報告書

出荷後の不具合原因(製品ベース)



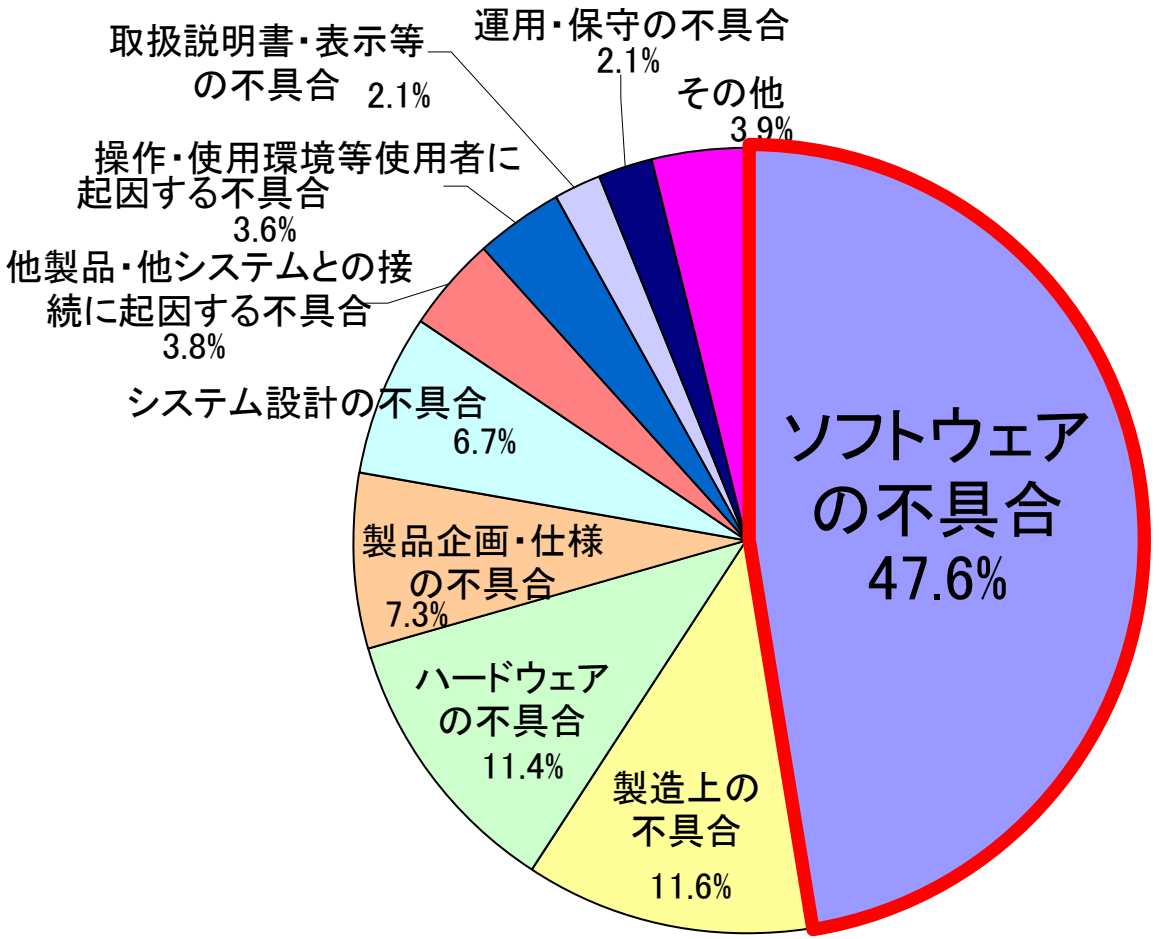
製品ベースでも、不具合原因で最も多いのがソフトウェア不具合

Copyright © 2013 IPA, All Rights Reserved

1．1　組込みシステムの不具合発生状況-3

2011年版組込みソフトウェア産業実態把握調査報告書

出荷後の不具合原因(件数ベース)



件数ベースでも、不具合原因で最も多いのがソフトウェア不具合

Copyright © 2013 IPA, All Rights Reserved

1．2　組込みソフトウェアのテスト実態調査結果

SEC Software Reliability Enhancement Center

問題（アンケートで多かった回答）

① • テストに多くの時間・コストがかかる → 価格競争力低下

② • 単体・結合テストが不十分 → 総合テストにしわ寄せ

③ • テストに頼った品質保証 → 設計を圧迫、テスト中心開発

④ • 過度の品質を求められる → 価格競争力低下

その他
- 先進的テスト手法の取り込み不足
- デグレードが発生する
- テストエンジニアの質が上がらない
- テストケースが作れない

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

課題（アンケート、ヒアリングの結果）

必要だが、無いと考えているもの

① ・テスト十分性を評価する尺度

1．『現実的』な品質とコストのバランスのとり方
2．開発『現場』の事情・状況に即した、
3．『実用的』な標準、基準とその策定の考え方
4．手法の『実用的』な活用方法、勘所

⑥ ・テスト工程毎の目的・範囲の標準

Copyright © 2013 IPA, All Rights Reserved

IPAへの期待（アンケート、ヒアリングの結果）

標準や基準と、その基本となる考え方が欲しい

汎用的な解はなさそう。
理論的な解はあっても、実用的な解はない。
　⇒他社はどうしているか、他社事例を参考にしたい。

自社の事例は出せないが、他社の事例は知りたい

自社のテスト基準や考え方を開示すると、
ネガティブキャンペーンに使われかねない
⇒他社とベンチマークして自社の妥当性を評価したい

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
2. **テスト事例集のコンセプト**
    1. **取り組むべき課題**
    2. **前提**
    3. **対象領域**
    4. **ESxRシリーズにおける位置付け**
3. テスト事例集の特徴
4. テスト事例集の構成
5. 利用方法
6. 事例紹介

Copyright © 2013 IPA, All Rights Reserved

２．１　取り組むべき課題について

SEC
Software Reliability Enhancement Center

- アンケート／インタビュー結果をIPA/SECにて分析・検討し、抽出した課題
  - テストの役割明確化
  - 時間・コストの認識の共有
  - 過度のテスト依存（弊害と限界の認識）からの脱却
  - テスト十分性の指標・基準値がない
  - 効率的かつ現実的なテスト手法・技法がない

しかし、課題は多岐に渡り、解決策も一様ではない。
汎用的なリファレンスガイドは馴染みにくい。

Copyright © 2013 IPA, All Rights Reserved

２．２　前提

現実的には、

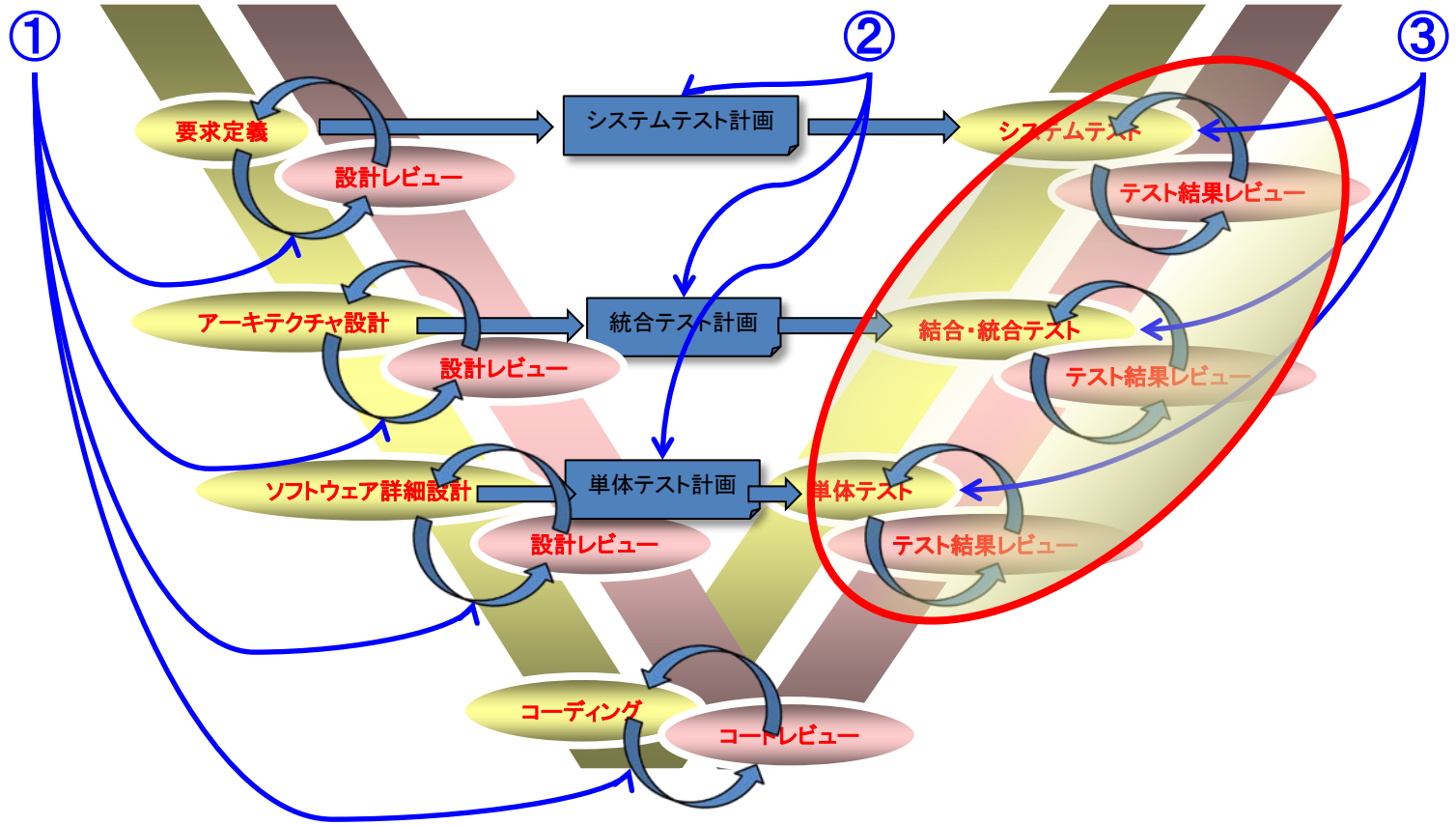
- いくらテストしても、バグ0にはならない
- テスト偏重による弊害がある
- 品質とコストのバランスが必要

この前提のもとに、
先進企業が実践する事例を収集し、
公開可能な表現に加工し、
事例集として発行することとした。

ベストプラクティス集ではない。
某社に100PJあり、その内たった一つのベストプラクティスを紹介してもベンチマークできない。残り99PJの実態を示す。

Copyright © 2013 IPA, All Rights Reserved

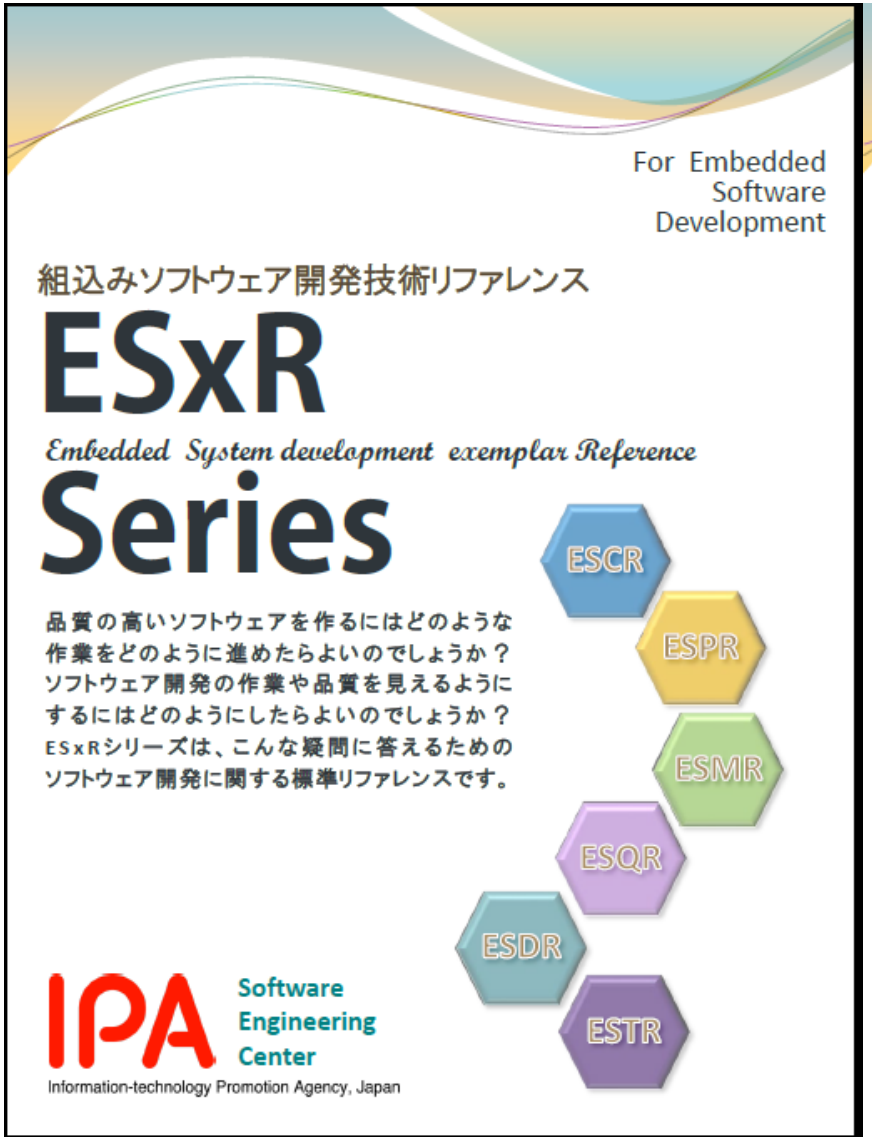
2．3　対象領域



W字モデルにおけるV&V（Verification & Validation）には①・②・③があるが、本書では狭義のテスト工程の③を対象とした

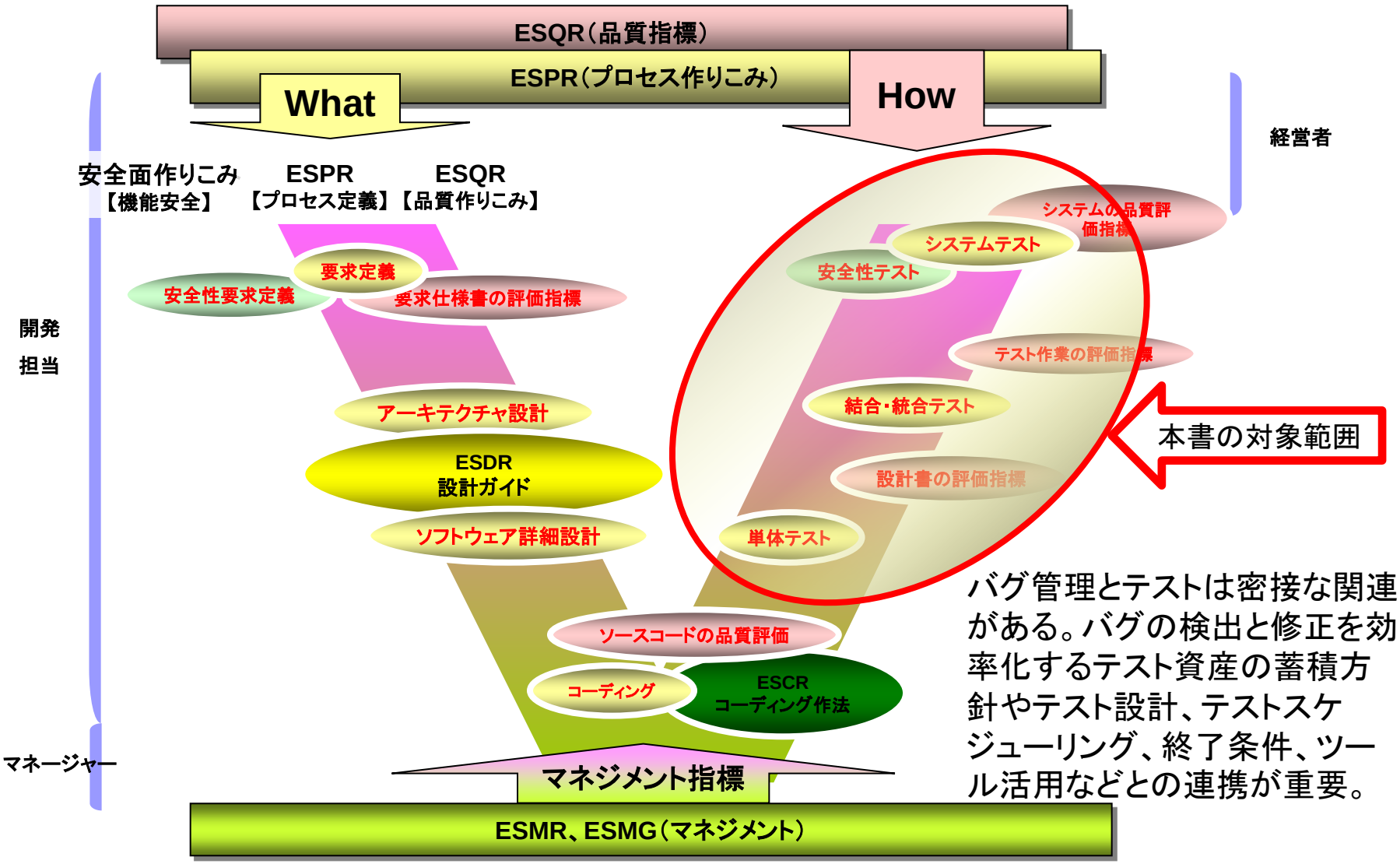
Copyright © 2013 IPA, All Rights Reserved

2．4　ESxRシリーズにおける位置付け



Copyright © 2013 IPA, All Rights Reserved

2．4　ESxRシリーズにおける位置付け



バグ管理とテストは密接な関連がある。バグの検出と修正を効率化するテスト資産の蓄積方針やテスト設計、テストスケジューリング、終了条件、ツール活用などとの連携が重要。

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
2. テスト事例集のコンセプト
3. **テスト事例集の特徴**
4. テスト事例集の構成
5. 利用方法
6. 事例紹介

Copyright © 2013 IPA, All Rights Reserved

３　テスト事例集の特徴

SEC
Software Reliability Enhancement Center

- 先進企業の実開発現場における成功･効果事例
- 組込みソフトウェアのテスト品質向上、テスト効率化の考え方･ノウハウを整理

- 実際の開発現場より成功･効果事例を収集して加工
- テスト指標･十分性判断基準･手法等を紹介（36事例）
- 実際の具体的事例を平易な解説で分かりやすく説明
- 定型フォーマット、Column、用語集付きで記述

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
2. テスト事例集のコンセプト
3. テスト事例集の特徴
4. **テスト事例集の構成**
    1. **3つのPartで36事例を解説**
    2. **目次**
    3. **各項の構成**
5. 利用方法
6. 事例紹介

Copyright © 2013 IPA, All Rights Reserved

4．1　3つのPertで36事例を解説

SEC Software Reliability Enhancement Center

【事例構成】3つの視点で36事例を分類

Part 1.　テストの役割と限界　19種

・テストの役割　（テストの実施方法、考え方・・・）
・テストの限界　（効率的なテスト方法・・・）
・基準値・目標値・終了条件　（尺度等の標準化・・・）

Part 2.　テストへの要求と対応状況　8種

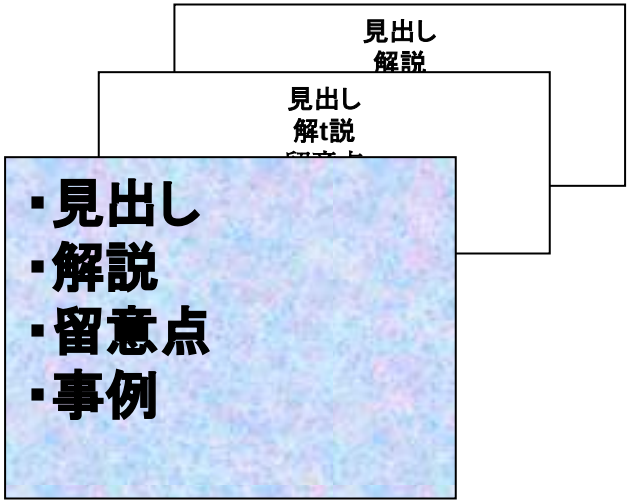
・ツール活用　（効率化のためのツール活用・・・）
・公的機関における基準　（公的機関基準等の活用）

Part 3.　テストの基本的テクニック　9種

・テスト技術・技法の分類　（項目十分性の追及・・・）
・環境　（性能測定環境不可・・・）
・教育　（ユーザ視点教育・・・）

【事例記述構成】

個別課題と対策事例を整理



Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

コンセプトシートの成果物イメージ案に沿った目次

Part 1 テストの役割と限界 9

1.1 テストの役割
- 1.1.1 テストの目的と位置付けの明確化 ……10
- 1.1.2 単体テスト容易性の向上 ……12
- 1.1.3 模擬環境と実機環境の相互補完 ……15
- 1.1.4 ソフトウェアタイプに応じたテスト項目チェックリスト ……18
- 1.1.5 結合テスト項目抽出のための３つの基本方針 ……21
- 1.1.6 差分開発での影響範囲解析への構造解析ツール活用 ……23

1.2 テストの限界
- 1.2.1 リグレッションテスト資産の蓄積と応用 ……25
- 1.2.2 重複、無駄、漏れを回避する総合テスト設計 ……27
- 1.2.3 戦略的テストスケジューリング ……29
- 1.2.4 コードカバレッジの目標と終了条件 ……32

1.3 基準値・目標値・終了条件
- 1.3.1 ソースコード行数の数え方 ……34
- 1.3.2 関係者との合意に基づく目標値設定 ……37
- 1.3.3 コストとのバランスを考慮したテスト項目抽出 ……39
- 1.3.4 総合テスト開始判断基準 ……41
- 1.3.5 段階付き合否判断基準 ……43
- 1.3.6 テストフェーズ移行判断基準の統一 ……45
- 1.3.7 達成不可能なコードカバレッジ基準とその終了可否判断 ……47

1.4 テスト管理
- 1.4.1 テスト効率向上のための戦略的テストシナリオ作成 ……49
- 1.4.2 テスト管理における問題の見える化 ……51

Part 2 テストへの要求と対応状況 55

2.1 ツール活用
- 2.1.1 テスト実行支援ツールの使い分け ……56
- 2.1.2 テスト管理ツールの統一 ……58
- 2.1.3 ツールのポータル化 ……61
- 2.1.4 ツールチェーン ……63
- 2.1.5 標準ツール選定とガイドライン制定 ……66
- 2.1.6 特徴の異なる静的解析ツールの使い分け ……67
- 2.1.7 Android 開発での模擬環境と実機環境の使い分け ……69

2.2 公的機関における基準
- 2.2.1 公的機関における基準 ……71

Part 3 テストの基本的テクニック 73

3.1 テスト技術・技法の分類
- 3.1.1 単体テストの観点に対応したテスト技法 ……74
- 3.1.2 同値分割、境界値分析手法の活用 ……76
- 3.1.3 加速度テストの目的と適用上の注意事項 ……78
- 3.1.4 合意に基づいた非機能テスト項目抽出 ……80
- 3.1.5 複数因子間組み合わせの勘所 ……82
- 3.1.6 モデル図を用いたテスト項目抽出手法 ……85

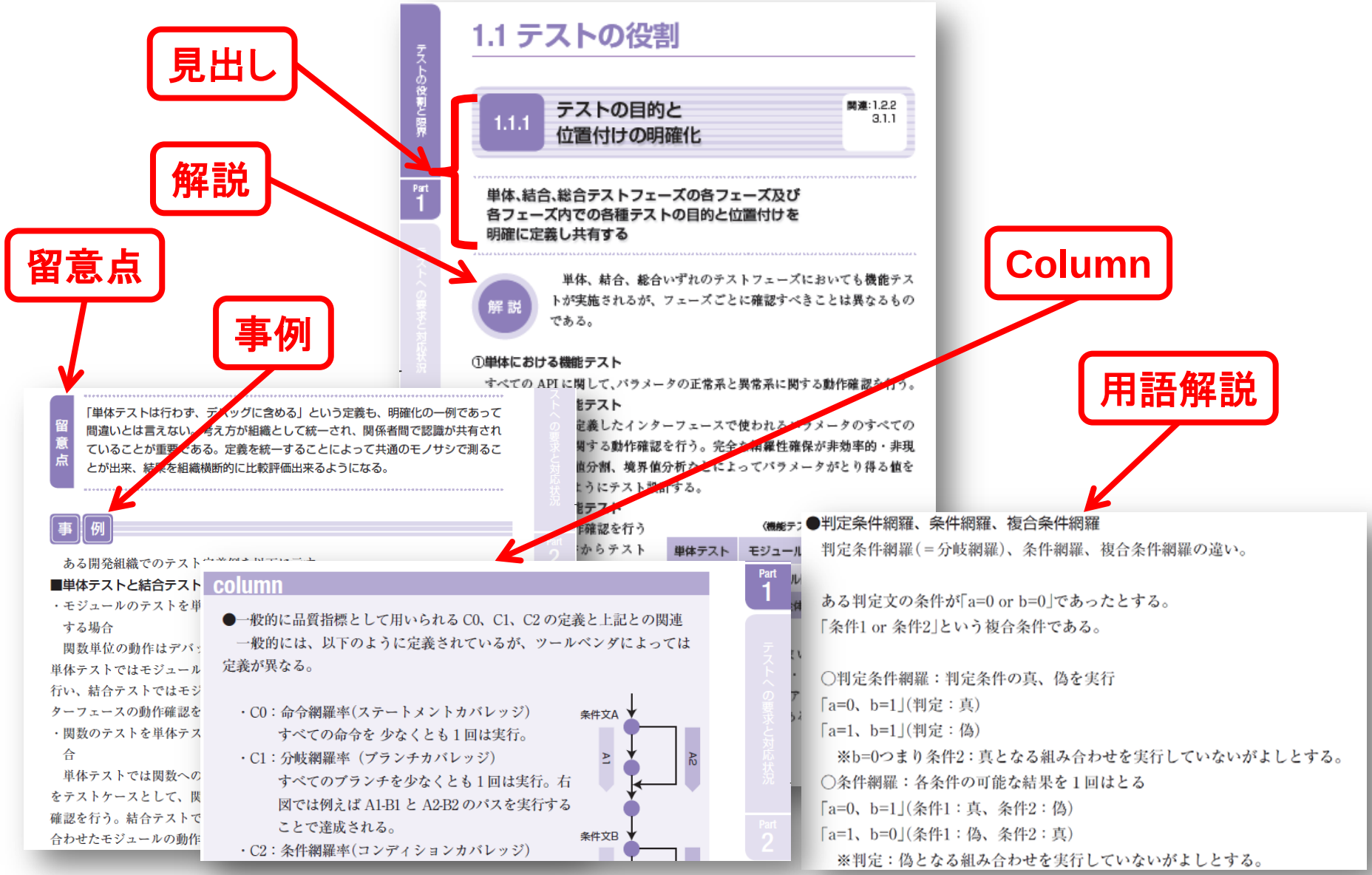
3.2 環境
- 3.2.1 性能評価環境がそろわない状況での性能評価 ……87

3.3 教育
- 3.3.1 テストに関する体系的教育カリキュラム例 ……89
- 3.3.2 テクニックと共に V&V の考え方の理解が必要 ……92

付録 ……93

Copyright © 2013 IPA, All Rights Reserved

4．3　各項の構成



Copyright © 2013 IPA, All Rights Reserved

4．3　各項の構成

SEC
Software Reliability Enhancement Center

見出し	各事例での対策における基本的な考え方、あるいは重要事項を端的に記述
解説	開発現場で実践されている事例の内容を図表などを用いて解説
留意点	留意すべき点、関連するトピックを記述
事例	実際の具体例をポイントを示しながら、公開可能な表現で記述
Column	事例解説に関連した事項を記述
用語解説	本書で頻繁に用いる用語や勘違いしやすい用語を解説

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
2. テスト事例集のコンセプト
3. テスト事例集の特徴
4. テスト事例集の構成
5. **利用方法**
6. 事例紹介

Copyright © 2013 IPA, All Rights Reserved

次のような場合に利用していただくことを想定しました

- 論理的に妥当性を証明し難い「当たり前」の裏付け
- 自社の場合と比較して、自社の取り組みを評価
- 自社の基準・標準を設定する際の参考
- 自社の基準・標準の妥当性を議論するきっかけ

事例に対して異論もあるかと思います。
「派生開発には使えない」、「PJ毎に事情が異なる」・・・

大いに議論していただけるきっかけになれば幸いです。

Copyright © 2013 IPA, All Rights Reserved

目次

1. テストの重要性と課題
2. テスト事例集のコンセプト
3. テスト事例集の特徴
4. テスト事例集の構成
5. 利用方法
6. **事例紹介**
    1. **Part 1　テストの役割と限界**
    2. **Part 2　テストへの要求と対応状況**
    3. **Part 3　テストの基本的テクニック**

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

Part 1 テストの役割と限界

- 1.2.3戦略的テストスケジューリング　IPA/SEC　石田
- 1.2.4 コードカバレッジの目標と終了条件　IPA/SEC 石田
- 1.3.1 ソースコード行数の数え方　IPA/SEC 石田
- 1.4.1 テスト効率向上のための戦略的テストシナリオ作成　NEC通信システム 羽田

Part 2 テストへの要求と対応状況

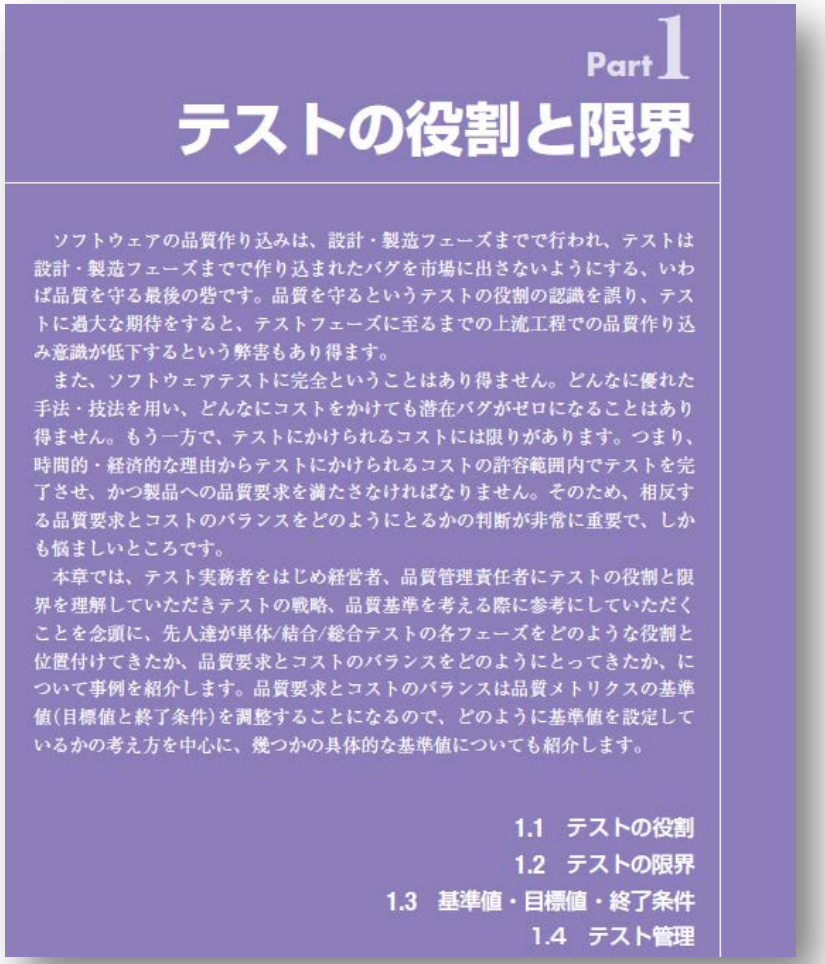
- 2.1.6 特徴の異なる静的解析ツールの使い分け　NEC通信システム 羽田
- 2.1.7 Android開発での模擬環境と実機環境の使い分け
- 2.2.1 公的機関における基準　NEC通信システム 羽田

Part 3 テストの基本的テクニック

- 3.1.3 加速度テストの目的と適用上の注意事項　IPA/SEC 石田
- 3.3.1 テストに関する体系的教育カリキュラム例　IPA/SEC 石田

Copyright © 2013 IPA, All Rights Reserved

Part 1　テストの役割と限界


Part 1
テストの役割と限界

ソフトウェアの品質作り込みは、設計・製造フェーズまでで行われ、テストは設計・製造フェーズまでで作り込まれたバグを市場に出さないようにする、いわば品質を守る最後の砦です。品質を守るというテストの役割の認識を誤り、テストに過大な期待をすると、テストフェーズに至るまでの上流工程での品質作り込み意識が低下するという弊害もあり得ます。

また、ソフトウェアテストに完全ということはあり得ません。どんなに優れた手法・技法を用い、どんなにコストをかけても潜在バグがゼロになることはあり得ません。もう一方で、テストにかけられるコストには限りがあります。つまり、時間的・経済的な理由からテストにかけられるコストの許容範囲内でテストを完了させ、かつ製品への品質要求を満たさなければなりません。そのため、相反する品質要求とコストのバランスをどのようにとるかの判断が非常に重要で、しかも悩ましいところです。

本章では、テスト実務者をはじめ経営者、品質管理責任者にテストの役割と限界を理解していただきテストの戦略、品質基準を考える際に参考にしていただくことを念頭に、先人達が単体/結合/総合テストの各フェーズをどのような役割と位置付けてきたか、品質要求とコストのバランスをどのようにとってきたか、について事例を紹介します。品質要求とコストのバランスは品質メトリクスの基準値(目標値と終了条件)を調整することになるので、どのように基準値を設定しているかの考え方を中心に、幾つかの具体的な基準値についても紹介します。

1.1　テストの役割
1.2　テストの限界
1.3　基準値・目標値・終了条件
1.4　テスト管理

テストは最後の砦

テスト偏重による弊害

限界：バグ0にはならない

限界：テストボリュームの爆発

限界：かけられるコストの制約

品質要求とコストのバランス

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

1.2.3 戦略的テストスケジューリング

関連：1.4.1

テスト効率向上のため異常系テスト項目も含め、過去のトラブルケースやブロッキングバグなど、リスクの高い項目から優先的に実施する。

解説

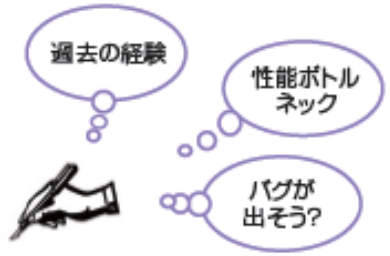
テスト優先順位の判断基準があいまいなまま、機能観点で大中小項目と機能分割してテストケースが書かれることが多い。別の観点でテストケースを書くことによって、少ないテストで大きな効果を出す方法がある。

組込み系システムでは、テスト全体に占める異常系テストの割合が高く、項目数も多い。また、異常系では発生条件をそろえるための準備に手間がかかるケースも多い。そのため、効率よく行うには、実施するべきテスト項目の優先付けが必要である。

少ないテスト件数で大きな効果を得る手法として、組み合わせテストでの組み合わせパターン数を間引く HAYST 法（直交表）、Pairwise（All-Pairs）法などがある。これらは製品特性に基づく主要な組み合わせを絞り込む方法で、その優先度付けを行う際の観点には以下のようなものがある。

①プラットフォーム及びこれに準じる共通的な機能のように、当該部分で生じるバグが全体のテスト進捗に影響を及ぼすと推測される部分を優先的にテストする。
②過去にバグが多発したモジュールから先にテストする。
③ブロッキングバグとなる可能性が高い（リスクが大きい）ものから先にテストする。
④基本機能は一通りテストする。
⑤性能のボトルネックになりやすい部位をテストする。

	a1	a2	a3	a4	a5	a6	a7	a8
b1	✔		✔					
b2	✔					✔		
b3		✔	✔					
b4				✔				
b5				✔		✔		
b6				✔				✔



また異常系のテスト項目抽出にあたっては、HAZOP (注) などの手法を活用するケースもある。

留意点

◆差分開発か、スクラッチ開発かによってもテスト実施優先度は異なる。差分開発では、主要機能は既に動作実績があるので、変更個所を優先して実施することが出来る。一方、スクラッチ開発ではすべてを新規に実装するので、キーになる機能、全体にかかわる部分などから着手することになる。

◆過去のプロジェクトのバグ分析からバグがありそうなポイントを推測することを推奨する、というのはよくあるが、実践するのは必ずしも容易ではない。現実的には、そういうポイントを勘と経験から見つけ出す達人に依存することが多い。論理的ではなくても、戦略的に優れた方法である。この場合、達人からの技術の継承が課題である。

Copyright © 2013 IPA, All Rights Reserved

事例紹介　1.2.3

SEC
Software Reliability
Enhancement Center

【事例1】以下に示すように、バグが発生する可能性が高いと思われるないしは、過去にバグが多く発生した機能部分については、テスト項目を重点的に追加する必要がある。

●リソース確保/開放のバグ発生の場合の実施例
- ・リソース確保/解放が確実に実施されることをテストする。
- ・リソース確保失敗のパターンをテストする。
- ・リソース確保成功と失敗が混在している場合、リソース確保成功の場合のみ開放されることをテストする。

●マルチタスクや割り込みバグ発生の場合の実施例
- ・データがプロセス/スレッドで共有されているかをチェックする。共有されている場合、そのデータに関してのアクセスパターンを分析しテストする。
- ・すべてのプロセス/スレッドについて、生成と消滅の組み合わせをテストする。
- ・出来るだけたくさんのプロセス/スレッドを立ち上げてテストを行う(プロセス/スレッドが突然終了する、もしくは終了させることが出来なくなる場合がある)。

【事例2】あるプロジェクトで実施された例。優先度を上げてテストすべき異常系テスト項目の対象となった過去のトラブルや、影響度の大きいバグが発生した際の具体的な対応内容を以下に示す。

●ブートプログラムの起動を行うことが出来ず、その上で動作するすべてのアプリケーションの確認が出来なくなる現象が発生した。

●アプリケーション動作に必要な情報を休止状態のようなステータスモードでいったんセーブしておき、起動時にそのデータを読み出すことでブート時間を短縮するといった起動高速化を実施している。この高速化処理では、目標時間内に完了出来るよう処理の短縮化を図る必要があり、このため、アプリケーション側のセーブデータ量をいかに少なく出来るかをトライ&エラーの試行錯誤で検討していたところ、アプリケーション側のデータ構造などにも影響が出てしまい、機能確認が出来ない状況になってしまった。

Copyright © 2013 IPA, All Rights Reserved

1.2.4 コードカバレッジの目標と終了条件

関連: 1.3.4 1.3.7

カバレッジ基準は、カバレッジ以外の指標や手段との併用など複合的視点に基づいて設定し、本来実行されるべきコードが実行されているかという観点でチェックする。

解説

大規模化し、複雑化した組込みソフトウェアでC0 100%は現実的でない場合が多い。一方、小規模でも割り込みを多用するOSやデバイスドライバでは、C0 100%は十分条件に成り得ない。いずれの場合もC0 100%は誤った目標ではないが、終了条件とすることには疑問がある。

例えば、C0カバレッジ(命令網羅)は原則としてテストツール計測による100%網羅を目標とするが、終了条件としては、実施コストとのトレードオフも勘案の上、目視によるコードレビューも含めて100%としても構わないとするのが現実的である。

また制御系機能に対してC1カバレッジ(分岐網羅)や状態遷移網羅テストを実施する場合においても、100%実施を必須とするかどうかについて同様の考え方があったとしても間違いではない。形式的な網羅度の達成を目指すよりも、本来実行されるべきコードが正しく実行されているかという観点でのチェックを行うことがより重要である。結果として、網羅率100%となることが理想である。

事例

ある組込み製品開発におけるカバレッジ基本方針の具体例を以下に示す。

【A社】

・網羅率が100%に満たない場合は、カバレッジ未達部分に対してコードインスペクションを実施する。

・C0:100%実施を目標とし、更に複雑度が高い場合はC1:100%実施を目標とする。

【B社】

・C0カバレッジ網羅:新規/修正した実装単位にすべてのソースコードを対象に確認する。

・C1カバレッジ網羅:新規/修正した実装単位にすべての分岐を対象に確認する。

・状態遷移網羅テスト:状態遷移モデルを特定した上で、すべての状態遷移を確認する。

・API網羅テスト:API仕様を特定した上で、すべてのAPIの全入力パラメータの正常値、両端値、異常値を確認する。

・C0カバレッジ、API網羅テストは基本的に必ず実施する。

・制御系機能に関するものは、C1カバレッジ、状態遷移網羅テストを実施する。

Copyright © 2013 IPA, All Rights Reserved

事例紹介　1.3.1

SEC
Software Reliability Enhancement Center

1.3.1 ソースコード行数の数え方

関連：1.3.2
1.3.4

テストに関するソフトウェア品質基準値の算出に行数を用いる際は、経験に基づき、実態を反映した行数見積もりとする

解説

テスト工数やテストで使用する品質基準値を算出する際、ソースコード行数をベースとすることは一般的に実施されている。また、修正モジュール全体への影響度合いや、再利用モジュールのテスト工数見積もりにおいては、対象製品やプロジェクトの特性によって係数を変えるなどの工夫も行われている。ただし、その都度適切な係数を決定するのも容易ではないという現実もあり、この重み付け係数を固定値として設定するケースも多い。

社内で係数を統一することにより、過去開発データとの比較が可能となることのメリットは大きい。

留意点

差分開発の場合、仮に新規開発の場合と開発対象行数が同じであったとしても、バグ密度の目標値を変えたり、行数の代わりに機能数やI/F数、モジュール数を分母にしたり、実態を反映するための補正を要する。また、機能数を用いる場合の補正では、経験の有無によって結果が大きく左右されやすい点に注意が必要である。

事例

差分開発、流用開発における例を以下に示す。

【A社の場合】

■既存資産に基づく差分開発、流用開発では、単純に追加・変更行数を足し合わせただけの行数ベースの見積方法では実態に合わないことが多い。例えば、行数が少なくても影響範囲が大きい場合などがこれに該当する。そこでソースコード行数算出式として以下のように重み付けした換算を行っている。

Σ(新規モジュール行数)＋0.5 Σ(修正モジュール行数)＋0.2 Σ(再利用モジュール行数)

■第2項は、修正行数だけではなく、修正が当該モジュール全体に影響を及ぼし得ることを意味する。

■第3項は修正なしで再利用するモジュールであっても、結合テストや総合テストの対象となるので、そのテスト工数分を反映しておく必要があるということを意味する。

【B社の場合】

■ソ

開発

新規

改造

追

造

そ

流用

■改

必要

更が

基準

■流

column

下記は、いずれの計測方法が正しいというものではなく、統一されていることが重要である。

・差分開発、流用開発、派生開発をする際、コンパイルオプションのSwitchでコンパイル対象から除外する部分の行数を含めるか否か。除外する場合、行数の計測にコンパイラのプリプロセッサなどのツールが必要になる。

・マクロ部分を、マクロ展開する前の行数とするか、マクロへの入力を反映させて展開したあとの行数とするか否か。

ソースコード行算出のモデル式としては、COCOMOⅡソフトウェア再利用モデル式というものもある。本式の係数導出には開発者の手間がかかるので、適用するならば開発者の負担増加(コスト増加)に見合う効果が期待できるか検討し、開発者に理解できるよう説明すべきである。

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

1.4.1 テスト効率向上のための戦略的テストシナリオ作成

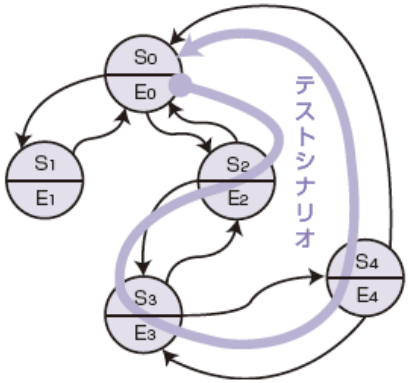
関連：1.2.1 1.2.3

組込みシステムでは異常系のテストケースが多く再現に手間がかかるケースも多いため、テスト項目同志の繋がりを考慮したテストシナリオを考えるなど効率よくテストするための工夫をする。

解説

組込みシステムの特徴として、ハードウェア異常検出やリトライ、リカバリなどの異常系処理が処理全体の大半を占めている。テストのためには、それら多くの異常状態を再現させる必要があるが、必ずしも再現は容易ではない。そこで、再現手順を容易にするための工夫と同時に再現手順の回数を削減させる工夫も、テスト効率化に有用である。

テスト実施に必要な状態とするための前処理に手間がかかったり、その状態にすること自体が困難なテストケースでも、テストシナリオによっては一連の作業の中で容易に作り出せることがある。テスト環境やテスト対象の状態がテストシナリオに合わせて変化していくので、シナリオに沿ってテストを行うことにより自ずと目的の状態になるというケースである。



また、あるテストの実行結果が次のテストのインプットになるなど、単独ではなく繋がりで考えることにより、少ないテストで効果が出せるケースもある。

テスト1によりinput2が自動的に生成されるのでinput2作成の手間が省ける

留意点

特に組込みシステムの異常系に関しては、状態作成が困難/手間がかかる場合が多い。そのため、必ずしも正常系と異常系とを分けてテスト実施するのではなく、正常系と異常系を組み合わせたシナリオとし、効率化することも検討すべきである。

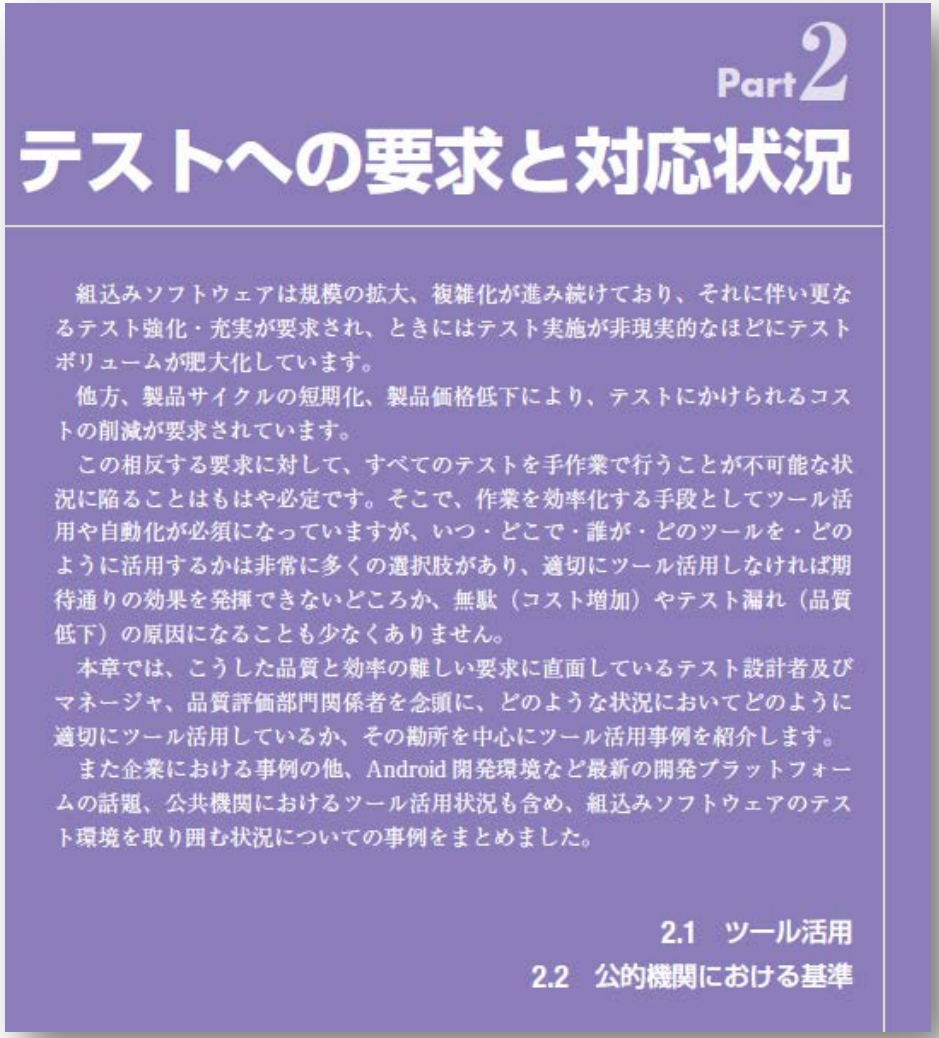
事例

以下に実際の開発組織で行われている取組み例を示した。これらの多くは、テスト可能状態にするために長時間を要したり手順が複雑であるような限界値テスト・異常テストケースである。とくに総合テスト段階で、その都度そうした手間をかけていることが現実的に困難であるという事情がある。

■メモリ容量に制限があるハンディ端末では、上限値に近い数 MB のデータを蓄積させ、その状態で更にデータを入れた場合の異常テストを行うが、この状況を生成することが容易ではないため、一連のテストシナリオの中に組み込んで、上限に近い状態で正常系も含めたテストを行う手順とする。

■フラッシュメモリのバンク切替えが必要になるような境界条件を作り出す限界テストを総合テストで行う際には、その状況を生み出す処理を他機能テストの連鎖で作り出せるようなテストシナリオを作成する。

Copyright © 2013 IPA, All Rights Reserved



Part 2

テストへの要求と対応状況

組込みソフトウェアは規模の拡大、複雑化が進み続けており、それに伴い更なるテスト強化・充実が要求され、ときにはテスト実施が非現実的なほどにテストボリュームが肥大化しています。

他方、製品サイクルの短期化、製品価格低下により、テストにかけられるコストの削減が要求されています。

この相反する要求に対して、すべてのテストを手作業で行うことが不可能な状況に陥ることはもはや必定です。そこで、作業を効率化する手段としてツール活用や自動化が必須になっていますが、いつ・どこで・誰が・どのツールを・どのように活用するかは非常に多くの選択肢があり、適切にツール活用しなければ期待通りの効果を発揮できないどころか、無駄（コスト増加）やテスト漏れ（品質低下）の原因になることも少なくありません。

本章では、こうした品質と効率の難しい要求に直面しているテスト設計者及びマネージャ、品質評価部門関係者を念頭に、どのような状況においてどのように適切にツール活用しているか、その勘所を中心にツール活用事例を紹介します。

また企業における事例の他、Android開発環境など最新の開発プラットフォームの話題、公共機関におけるツール活用状況も含め、組込みソフトウェアのテスト環境を取り囲む状況についての事例をまとめました。

2.1　ツール活用

2.2　公的機関における基準

テストボリューム肥大化

テスト期間短縮要求

ツールによる自動化・効率化

ツール導入時の課題

Android開発のテスト環境

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

2.1.6 特徴の異なる静的解析ツールの使い分け

関連: 1.1.6/ 2.1.1/2.1.2/ 2.1.3/2.1.4/ 2.1.5

静的解析ツールを用いる場合、ツールの特徴を把握すること。そして、開発のどのフェーズで用いるか、どういう問題の検出を期待するかによって、いつ・誰が・どのツールを用いるかを決定する。

解説

静的解析ツールを適用する場合、ツールが検出可能な欠陥種別など、特徴の異なる複数の静的解析ツールを適用することも検討する。

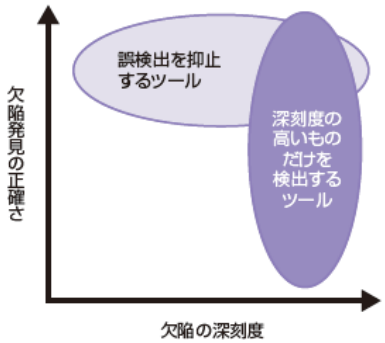
例えば、以下のような3つのツールを適用する。

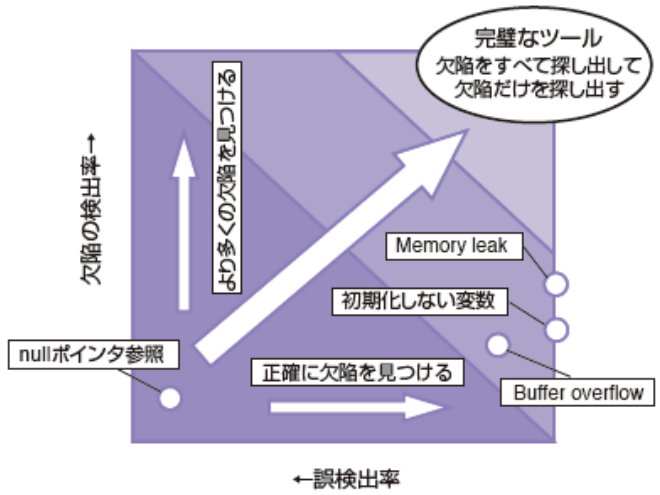
〔ツール1〕比較的短時間で大規模なコードベースにおける一般的なコーディング欠陥や疑わしいコーディングパターンを発見するのを得意とするもの。

〔ツール2〕同様のツールに比べて終了までの時間が長くかかるが、より深いパス解析を必要とするあいまいな欠陥や疑わしいコーディングパターンを発見できるもの。

〔ツール3〕プログラムにおける主に3つの最も一般的なタイプのソフトウェアの欠陥(初期化されていない変数、nilポインタの逆参照、範囲外配列インデックスの使用)を早く簡単に発見できるもの。

それぞれのツールの特徴を考慮し、適用する時期を検討するとよい(コーディング終了後、レビュー前、チェックイン前、単体テスト実行前、外部チームからの受け入れ時、など)。





静的解析ツールの特徴を把握する

留意点

一般的な静的解析ツールには、ツールが検出すべき問題点を定義するデータベースがあり、検出数を調整する機能がある。しかし、それでも静的解析ツールでの誤検出は避けられないので、誤検出抑止のためのフィルター/ラッパーを自部門向けに作成して、静的解析ツールと組み合わせて使うとよい。

Column

複数の静的解析ツールを使い分けるというのは、ツールの導入費用や実行時間などに厳しい制約が無い場合に可能なことであって、自分達でツールを準備、あるいは実行できない場合は、第三者によるCDI(Code Inspection)サービスの利用も検討するとよい。

Copyright © 2013 IPA, All Rights Reserved

事例紹介　2.1.7

SEC
Software Reliability
Enhancement Center

2.1.7 Android開発での模擬環境と実機環境の使い分け

関連: 1.1.3
3.2.1

シミュレーション環境と実機環境でのテストケースを切り分ける。

解説

Androidには豊富な開発環境が整備されている。テスト環境としては、OSS(オープン・ソース・ソフトウェア)で高速なバイナリシミュレータQemu(キューエミュ)を用いた仮想HW(goldfish)が用意されており、PC上で製品としてのシステムシミュレーションが可能になっている。

goldfishは、基本HWのモデルをそろえている。その他の周辺HWについては、利用者がモデルを作成する。そのためのツールとして、Android仮想デバイス(Android Virtual Device : AVD)の設定をサポートしている。

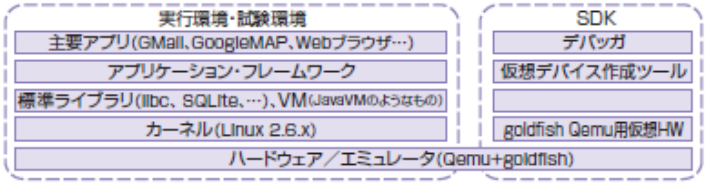
ただし、goldfishによるシミュレーション環境は、実機用のソフトウェアをそのまま使用出来るので機能面はかなり忠実に(低レベルの抽象度で)模擬するが、タイミングについては実機のタイミングを模擬していない。

つまり、機能テストについては、製品を構成する部品がそろった環境で実機用のカーネルから、デバイスドライバ、ミドルウェア、アプリまでを統合してテスト出来るので、ほとんどのテストケースを実行することが出来る。とは言っても、製品の外部については適切なモデルを作成して組み込む必要があり、それはたやすいことでないことに注意すべきである。

一方、タイミングに関するテストは、実機での実行結果と異なるため、タイミングに依存するテストケースは実機環境で行わなければならない。実機環境で実施するためのテストプログラムの検証にシミュレーション環境を使用することによって、実機環境使用時間を短縮出来る。

シミュレーション環境で出来るテストケース、実機で実施すべきテストケース、実機使用前にシミュレーション環境で準備出来る作業を事前に切り分けて、以下のことを心がけるべきである。

①無意味なテストを実施しない(不適切なテスト環境でのテスト実施排除)。
②同じテストケースの重複テスト実行を避ける(無駄を排除)。
③実機環境での作業時間を最小にする(テスト効率化)。



Android実行環境と統合開発環境

留意点

製品の外部については適切な抽象度のモデル(外界モデル)を作成して組み込む必要があり、それはたやすいことではない。テスト対象が何であるかを明確にした上で、作成するモデルの抽象度を決定する。いたずらに抽象度を下げて現実の動作を、仕組みまで含めて忠実にモデリングすることが優れたモデリング技術ではないことを理解すべきである。

事例

Qemuを用いたシステムシミュレーションは、タイミングを模擬出来ないという短所があるが、Qemuにアドオンする形で実機と同じタイミングを再現する方法について、以下に示す。

■論文「QEMUによるHW/SW 協調シミュレータの構築」(設計・検証技術、一般セッション、アーキテクチャ、情報処理学会創立50周年記念)の方法によると、ソフトウェア内部のタイミング(命令レベルの時間精度)を保証し、かつ周辺ハードウェアモデルに設計通りのタイミングを再現させることが出来るため、タイミングに依存するテストケースについても、かなりの部分を実機レスでテストすることが出来る。この方法は実機と同等の実行性能であることも大きな特徴である。

■単純にタイミングを再現するだけであれば、それほど高度な技術を必要としないが、実行性能が実機の何百倍も遅かったら、ソフトウェア検証用として用をなさない。ハードウェア検証用シミュレータには、この手のものが多いので、シミュレータを選択するときには、ハードウェア検証用シミュレータとソフトウェア検証用シミュレータの違いを意識する必要がある。前述の論文では、高速なソフトウェア検証用シミュレータとしてQemuを用いた。

column

CPU動作をモデル化することによって、"コンピュータを使ってコンピュータをシミュレートする"ものを、エミュレータあるいは命令セットシミュレータ(ISS : Instruction Set Simulator)と呼ぶ。Qemuはこれに属する。

一般的にISSでは高速化のため、1命令ごとにシミュレートするのではなく、命令ブロック(ベーシックブロックとも呼ぶ)単位でシミュレートする。

Copyright © 2013 IPA, All Rights Reserved

2.2.1 公的機関における基準

関連: 1.3.3 2.1.1 2.1.6

高度な安全性を要求される製品に対する、公的機関による安全性検証の例

解説

開発対象システムのソフトウェア不具合が甚大な人的・経済的損失を引き起こすような問題を特定し、その問題の有無をテストする場合には、様々な観点でのテストが必要になる。

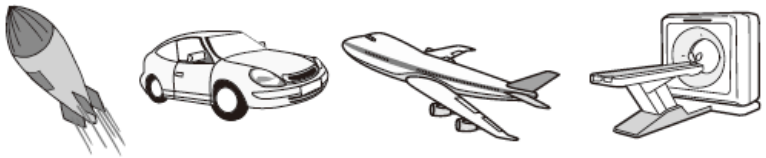
例えば、以下のような観点でのテストをそれぞれ実施することが考えられる。

①コーディングの欠陥(実装)
②アルゴリズムの欠陥(設計・ロジック)
③タスクの干渉(競合状態、データ破損)
④故障保護の不足

ただし、テスト対象とする問題を事前に特定することが重要であり、問題を特定せずに、これらすべてのテストを実施しても効果が期待出来ず、コストやスケジュールを無駄に消費する可能性が高い。

また、どこまでコストをかけて行うかは、製品に要求される安全性・信頼性の程度によって判断する必要がある。

公的機関がどのように、どこまで検証して問題の有無を判断したかは、参考になる。



ソフトウェアの不具合が甚大な人的・経済的損失を引き起こしかねない組込みシステム

留意点

様々な観点から多くの検証を行うには、ツールを活用して効率化を図る必要がある。その際、ツールには得手・不得手があるため、複数のツールを組み合わせることが有効である。

事例

検証を行っている公的機関として、例えばNASAでは、実際に以下のような分析・検証を行い、ソフトウェア上の欠陥が見つからなかったことを報告している。

1.コーディングの欠陥(実装)

静的ソースコード解析ツールを使用したソフトウェア実装解析にそれぞれ特徴の異なる3種類のツールを使用。

2.アルゴリズムの欠陥(設計・ロジック)

モデルベース設計ツールを使用したソフトウェアのアルゴリズム設計分析を実施。

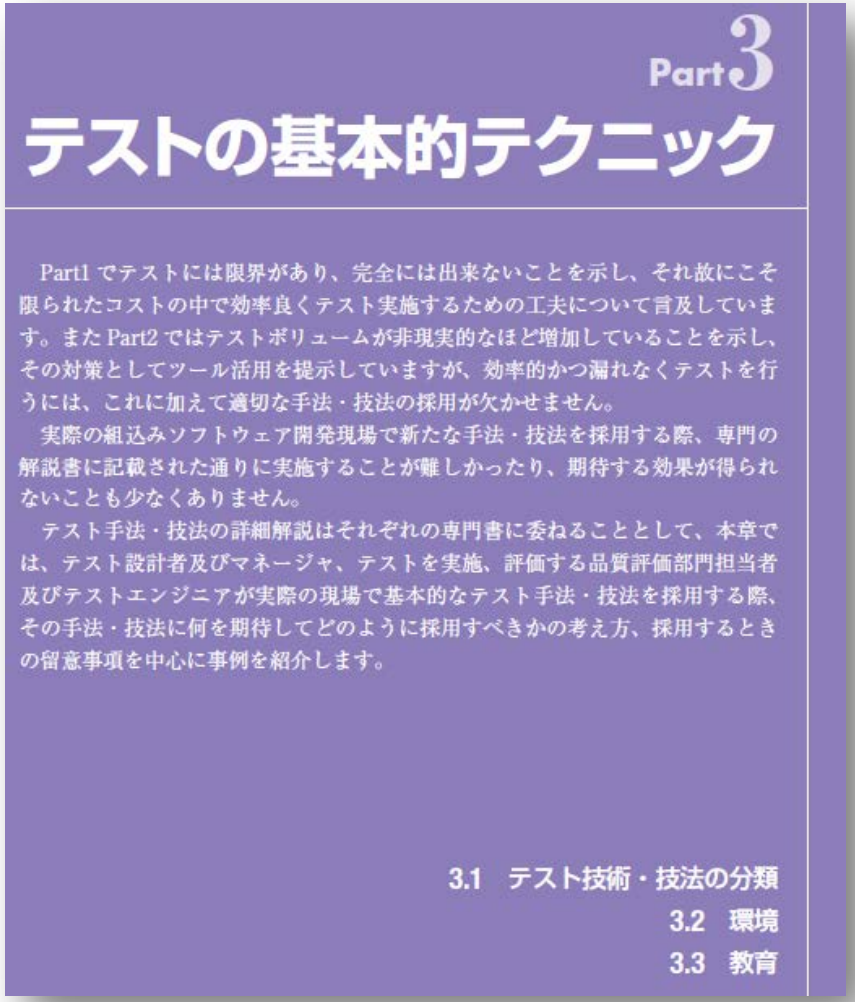
3.タスクの干渉(競合状態、データ破損)

ツールを用いた設計分析を行っている。SPINなどのモデル検査ツールを使用したソフトウェアロジックモデルのチェック。

4.故障保護の不足

ロジックモデルのチェックとして、問題発生条件の組み合わせテストを行った。FTAなどの分析手法を用いて故障保護の不足が無いかの検証を実施。

Copyright © 2013 IPA, All Rights Reserved


Part 3
テストの基本的テクニック

Part1 でテストには限界があり、完全には出来ないことを示し、それ故にこそ限られたコストの中で効率良くテスト実施するための工夫について言及しています。また Part2 ではテストボリュームが非現実的なほど増加していることを示し、その対策としてツール活用を提示していますが、効率的かつ漏れなくテストを行うには、これに加えて適切な手法・技法の採用が欠かせません。

実際の組込みソフトウェア開発現場で新たな手法・技法を採用する際、専門の解説書に記載された通りに実施することが難しかったり、期待する効果が得られないことも少なくありません。

テスト手法・技法の詳細解説はそれぞれの専門書に委ねることとして、本章では、テスト設計者及びマネージャ、テストを実施、評価する品質評価部門担当者及びテストエンジニアが実際の現場で基本的なテスト手法・技法を採用する際、その手法・技法に何を期待してどのように採用すべきかの考え方、採用するときの留意事項を中心に事例を紹介します。

3.1 テスト技術・技法の分類
3.2 環境
3.3 教育

適切な手法・技法の採用

手法・技法採用時の留意事項

テストエンジニア育成

Copyright © 2013 IPA, All Rights Reserved

事例紹介　3.1.3

Typo
“対抗”は“対向”の誤りです。

SEC
Software Reliability Enhancement Center

3.1.3 加速度テストの目的と適用上の注意事項

関連: 1.1.3 1.2.1

加速度テストの実施にあたっては、目的に対する適用妥当性や適用上の限界も考慮する。

解説

ここではソフトウェアの加速度テストについて、その目的や適用上の注意事項などについて解説する。

①目的

加速度テストは以下のような目的で実施される。

・テスト時間を短縮し、また短時間でテスト網羅度を向上させる。
・過負荷状態のテスト環境を再現する。
・実機で所定期間内にテストするには非現実的なほどに長時間を要する。
・リグレッションテストでのテスト網羅度を上げてデグレードを回避する。

②分類

加速する対象として、テスト対象そのものを加速するケースとテスト対象の周辺要素を加速するケースがある。

前者の事例として、通信機器での対抗装置からの応答を早めること、すなわち、対抗装置が実際の処理応答をはるかに超えるような高速処理状態を再現するケースが該当する。また後者の事例としては、携帯電話のGUI操作のテストを実施する場合に、通話やメールの受信を超高頻度に発生させるようなケースがある。

③注意事項・適用限界

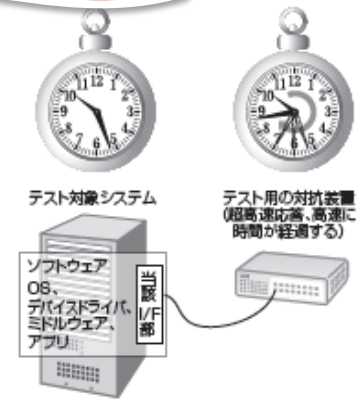
加速度テストの適用にあたっては、次のような点に注意が必要である。

・物理現象そのものは加速出来ない

例えば、冷蔵庫の温度設定を変えることなく、製氷機で氷が出来る時間を短縮することは出来ない。温度計の計測値下降を加速することは出来ても、実際に温度が下がらなければ水は凍らない。

・内部(テスト対象)と外部(対抗)の時間経過が異なる

対抗装置からの応答を速めることによって、テスト対象モジュール(インターフェース部)の高負荷状態を再現することは出来る。しかし、テスト対象システムの時間が早く経過するわけではなく、対抗装置の時間だけが早く経過している。テスト対象システムでは対抗とのインターフェース部が高負荷になるのみで、他の部分は高速処理もしていない。よって、テスト対象モジュールを含むテスト対象システムの動作が長時間経過することによって発生するメモリーリークを短時間で再現することは出来ない。



留意点

◆外部環境モデルまでを含めて時間精度保証が可能なSILSシステムシミュレータであれば、システム全体の時間経過を早めることが出来、通常動作が長時間経過した場合に発生するメモリーリークについてもかなりの精度で再現することが出来る。

◆ただし、現状は以下のような事情から、実用レベルの製品開発で適用されているケースは多くない。

・システムシミュレータは高価である
・外界(外部環境)モデルまで含めたシステムのモデリングには、高度なモデリング技術が必要である
・時間精度を保証(≒タイミングを再現)しつつ高速実行可能なシステムシミュレータは少ない
・外界モデルまで含めて時間精度を保証可能なシミュレータは少ない

事例

加速度テストの適用例及び今後の可能性について以下に示す。

■加速度テストを実施している例としては、以下のようなものがある。

・デジカメなどのバッテリー残量の減少速度を加速する。
・炊飯器の温度調整をタイマー調整し時間短縮して実施する。

■今後、加速度テストの適用が有用と思われるケースとしては、プリンタのインク量の模擬がある。インク量が多→少となるスレッシュホールドを通過するケースを通常のやり方で模擬しようとすると長時間を要するため、加速度テスト適用が有効であろう。

Copyright © 2013 IPA, All Rights Reserved

SEC Software Reliability Enhancement Center

3.3 教育

3.3.1 テストに関する体系的教育カリキュラム例

関連:

ソフトウェアの開発やテストにかかわるエンジニアのスキル向上と平準化に向けた、体系的な取り組みを行う。

解説

総合テストやソフトウェア総合テストは第三者テストの観点から開発から独立したテストメンバにより行われるなど、開発の各工程で設計や品質保証など様々なエンジニアがテスト行為に携わる現状に鑑みると、ソフトウェア技術者全般を対象としたテストの手法、技法の知識習得教育を組織として体系的に実施することには大きな意義がある。

こうした教育の全体像を考えるにあたっては、テストエンジニアがどういうスキルを持つべきかの方針を明確にし、体系的にカリキュラムを考える必要がある。

また、現場ですぐに実践出来るよう、座学だけでなく演習も含めた実践的スキルの習得が出来るようなカリキュラムの工夫も必要であろう。

講座名	内容
ソフトウェアシステム概論	ソフトウェアシステムを取り巻く状況概論を解説。
テストの基礎	テストとは何か、なぜテストを行うのか、V字モデルとテストフェーズなどを解説。
テスト技法	同値分割法、境界値分析、デシジョンテーブル、状態遷移テストなどの技法を解説。
ソフトウェアメトリクス	ソフトウェアの規模、開発工数、開発期間、ソフトウェア欠陥率、ソフトウェア欠陥除去率、サイクロマティック複雑度、コードカバレッジなど品質指標を解説。

留意点

◆コアアーキテクチャは一度設計すると長年変更しないことが多く、設計者が定年退職してしまい、新人など次世代がそのノウハウや技術を継承出来ないというケースも現実に発生している。

◆そこで、近年では新人や中堅向けにコアアーキテクチャの教育を社内で実施する取り組みを行う企業も出てきた。現業の仕事はすべてキャンセルし２週間程度の集中研修を受講させることにより、技術の底上げに効果が出始めている。

事例

ある組織で運用中の個別カリキュラム例を以下に示す。

■単体テストのテスト項目設計のテクニックとテストツール活用のテクニック

不要な単体テスト項目を少なくするコーディング、ユニットテストツールの実装など、テストの効率化を意識した、実践的テクニックの知識を習得する。

■テスト技法

高品質のソフトウェアを製作するために、ソフトウェア開発工程全体で適切なテスト技法を用いた品質の作り込みを体得する。

■ソフトウェア品質保証

各種ソフトウェア品質保証技術、プロセス改善技術が編み出されてきた経緯を理解し、効果的にソフトウェア品質を向上させるためのポイントの本質を理解するなど、基本的な概念の理解から実践的技術活用までの知識を習得する。

■ソフトウェア品質管理と品質管理データの活用

実践的なソフトウェア品質管理とその運用方法を理解する必要がある。ソフトウェアの品質管理に必要な品質データをタイムリーに収集し、統計技法など各種技法を用いて実態を分析/評価し、PDCAでの運用実践するための知識を習得する。

■テスト計画の立て方、テスト戦略

効率的にテストを行い、品質を確保するためのテスト計画やテスト戦略の立案などテストチームリーダとして必要な能力を養成する。

■ SEPG/SQA 担当開発プロセススペシャリスト養成講座

ソフトウェア品質向上に向けた、開発プロセスの見える化と改善を図るために必要なスキルや管理方法を実践活動の中で体得する。品質管理計画、マスターテストプラン立案などの実習を含む。

Copyright © 2013 IPA, All Rights Reserved

事例紹介　3.3.1

SEC
Software Reliability
Enhancement Center

組込みソフトウェア技術者向け教育メニューの具体例
C言語/C++言語/Java言語
MDA技術講座(Rhapsodyを用いたオブジェクト指向開発)
Perl入門
RTOS基礎/応用
UMLによるCプログラミング/C++プログラミング/Javaプログラミング
UMLモデリング
UML分析モデリング基礎(分析モデル作成のための基礎トレーニング)
UML分析モデリング実践(オブジェクト指向分析/設計講座)
アセンブラ基礎/応用
オブジェクト指向とUML入門
コーディング品質向上
ソフトウェアテスト技法基礎
ソフトウェアドキュメンテーション講座
ソフトウェア開発の構成管理と変更管理入門
デジタル回路入門
マイコンハードウェア入門
レビュー技法
構造化モデリング
設計モデリング・実践(C言語)
設計図活用
組込みC言語基礎1(ポート制御とビット演算)
組込みC言語基礎2(割込み制御)
組込みC言語基礎3(タイマ制御)
組込みC言語応用(チャタリング除去)
組込みLinuxアプリケーションプログラミング基礎
組込みLinuxデバイスドライバプログラミング基礎
組込みソフトウェアの代表的な詳細設計法
組込みソフトウェアの方式設計と実装法

Copyright © 2013 IPA, All Rights Reserved

ご清聴ありがとうございました

Copyright © 2013 IPA, All Rights Reserved